Method for segmentally deinterleaving a data signal

Patent number: DE10048872
Publication date: 2002-04-25

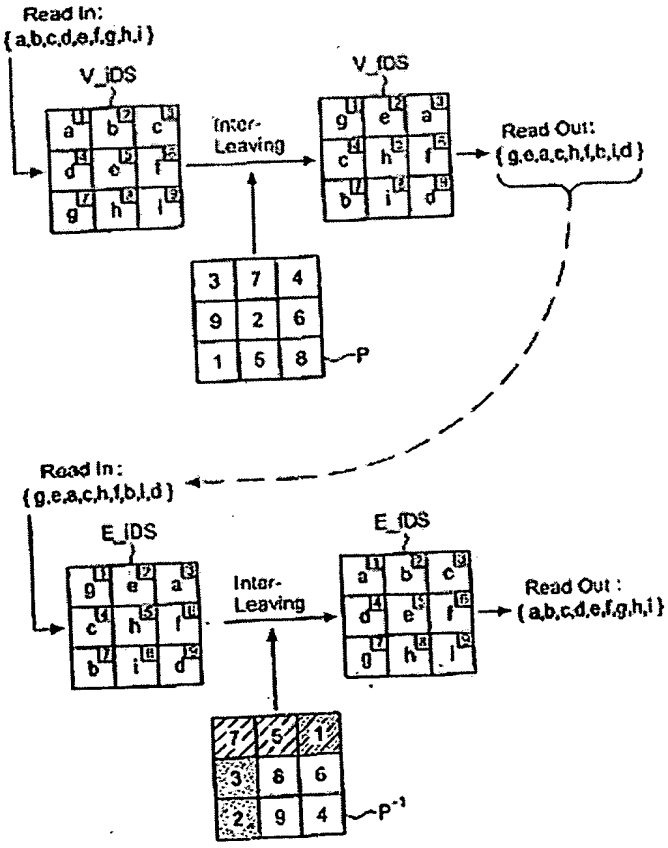
Inventor: SCHNEIDER MICHAEL (DE); BECKER BURKHARD (DE); DOETSCH MARKUS (DE); JUNG PETER (DE); KELLA TIDEYA (DE); PLECHINGER JOERG (DE)
Applicant: INFINEON TECHNOLOGIES AG (DE)

Classification:
- international: H03M13/29
- european: H03M13/25T, H03M13/27, H04L1/00B3, H04L1/00B5T, H04L25/03E3
Application number: DE20001048872 20001002
Priority number(s): DE20001048872 20001002

Also published as:
W O0230073 (A3)
W O0230073 (A2)
US 2003221157 (A1)

Abstract not available for DE10048872
Abstract of correspondent: **US2003221157**

In a method for deinterleaving a data signal interleaved in blocks in accordance with a prescribed interleaving specification, deinterleaving target addresses are calculated for a first prescribed segment of the data symbols to be deinterleaved, and are stored in a target address memory. The relevant segment of the data symbols is then deinterleaved by using the calculated target addresses. Subsequently, these two steps are repeated until the entire data block has been segmentally deinterleaved.



Data supplied from the ***esp@cenet*** database - Worldwide

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

Offenlegungsschrift

(10) DE 100 48 872 A 1

(51) Int. Cl.[7]: H 03 M 13/29 // (H04B 7/216,H04Q 7:20)

(21) Aktenzeichen: 100 48 872.2
(22) Anmeldetag: 2. 10. 2000
(43) Offenlegungstag: 25. 4. 2002

(71) Anmelder:
Infineon Technologies AG, 81669 München, DE

(74) Vertreter:
Patentanwälte Dr. Graf Lambsdorff & Dr. Lange, 81673 München

(72) Erfinder:
Schneider, Michael, 56823 Büchel, DE; Jung, Peter, 67697 Otterberg, DE; Plechinger, Jörg, 80469 München, DE; Kella, Tideya, 80337 München, DE; Doetsch, Markus, 56072 Koblenz, DE; Becker, Burkhard, 85737 Ismaning, DE

(56) Entgegenhaltungen:

DE	198 46 721 A1
US	60 29 264
US	56 59 580

Raouafi F., Dingninou A., Berrou C.: "Saving memory in turbo-decoders using the Max-Log-MAP algorithm" IN IEE (1999);

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Abschnittsweise Entschachtelung

(57) Bei einem Verfahren zur Entschachtelung eines gemäß einer vorgegebenen Verschachtelungsvorschrift blockweise verschachtelten Datensignals werden Entschachtelungs-Zieladressen bezüglich eines ersten vorgegebenen Abschnitts der zu entschachtelnden Datensymbole berechnet und in einem Zieladressenspeicher abgelegt. Mittels der berechneten Zieladressen wird dann der betreffende Abschnitt der Datensymbole entschachtelt. Nachfolgend werden diese beiden Schritte so oft wiederholt, bis der gesamte Datenblock abschnittsweise entschachtelt ist.

DE 100 48 872 A 1

DE 100 48 872 A 1

Beschreibung

[0001] Die Erfindung betrifft ein Verfahren zur Entschachtelung eines blockweise verschachtelten Datensignals.

[0002] In der Telekommunikationstechnik ist es üblich, ein über einen Kanal zu übertragendes Datensignal einer senderseitigen Verschachtelung zu unterziehen. Durch die Verschachtelung wird erreicht, daß Störungen, die ohne eine Verschachtelung statistisch abhängige (in Gruppen auftretende) Detektionsfehler bewirken würde, stattdessen statistisch unabhängige Detektionsfehler erzeugen. Für statistisch unabhängige Detektionsfehler ist durch Kanalcodierung ein besserer Fehlerschutzgrad als für statistisch abhängige Detektionsfehler erreichbar.

[0003] Die Ver- und Entschachtelung des Datensignals erfolgt Datenblockweise, d. h. Datenblock für Datenblock wird nach einer jeweils gleichen Verschachtelungsvorschrift vom senderseitigen Verschachteler verschachtelt und nach der inversen (ebenfalls jeweils gleichen) Entschachtelungsvorschrift vom Entschachteler im Empfänger entschachtelt.

[0004] Zu diesem Zweck müssen vor der ersten Ver- bzw. Entschachtelung die entsprechenden Zieladressen (Verschachtelungs-Zieladressen bzw. Entschachtelungs-Zieladressen) für die Umsortierung der Datensymbole berechnet werden. Bisher geschieht dies so, daß vor Durchführung der ersten Ver- bzw. Entschachtelungsprozedur Zieladressen für sämtliche Datensymbole eines Datenblocks berechnet und in einem Verschachtelungs-Zieladressenspeicher bzw. Entschachtelung-Zieladressenspeicher abgelegt werden. Bei einem Datenblock bestehend aus K Datensymbolen müssen die Zieladressenspeicher jeweils K Zieladressen-Speicherplätze umfassen. Die Zieladressenspeicher enthalten somit die kompletten Ver- bzw. Entschachtelungsinformationen.

[0005] Nachteilig bei diesem Verfahren der Entschachtelung ist, daß ein großer Speicherplatzbereich im Empfänger eingerichtet sein muß. Für den UMTS-(Universal Mobile Telecommunications System-)Standard, der eine Datenblocklänge K zwischen 40 und 5114 Bits erlaubt, wird für die Abspeicherung der Entschachtelungs-Zieladressen ein Speicher mit 5114 Speicherzellen einer Adreßdatenbreite von 13 Bit benötigt.

[0006] Üblicherweise erfolgt die Verschachtelung eines Datensignals nach der Kanalcodierung. Bei einer speziellen Form der Kanalcodierung, die als Turbo-Codierung bezeichnet wird, wird jedoch bereits bei der Kanalcodierung eine Verschachtelungsprozedur durchgeführt. Diese im Rahmen der Turbo-Codierung durchgeführte Verschachtelung wird im folgenden als Turbo-Verschachtelung bezeichnet.

[0007] Turbo-Codes sind binäre, parallel verkettete, rekursive, systematische Faltungscodes. Durch die Verwendung von Turbo-Codes läßt sich insbesondere beim Übertragen von großen Datenblöcken bestehend aus mehr als z. B. 1000 Bits ein erheblich besserer Fehlerschutzgrad als mit üblichen Faltungscodes erzielen. Die Struktur eines Turbo-Codes sowie die Erzeugung desselben unter Verwendung eines Turbo-Codierers mit integriertem Turbo-Verschachteler sind bekannt und beispielsweise detailliert in dem Buch "Analyse und Entwurf digitaler Mobilfunksysteme", von P. Jung, Stuttgart, B. G. Teubner-Verlag, 1997, Anhang E, Seiten 343–368, beschrieben.

[0008] Beim Empfang des Turbo-codierten, über einen Übertragungskanal (z. B. Mobilfunkkanal) übertragenen Datensignals muß im Empfänger im Zuge der Turbo-Decodierung auch die Turbo-Verschachtelung rückgängig gemacht werden. Dieser Prozeß wird als Turbo-Entschachtelung bezeichnet und durch einen im Turbo-Decodierer integrierten Turbo-Entschachteler bewerkstelligt. Die Turbo-Ver- und Entschachtelung des Datensignals erfolgt ebenfalls Datenblockweise.

[0009] Der Erfindung liegt die Aufgabe zugrunde, ein verbessertes Verfahren zur Entschachtelung, insbesondere Turbo-Entschachtelung, eines blockweise verschachtelten Datensignals anzugeben. Insbesondere soll das Entschachtelungsverfahren einen möglichst geringen Speicherplatzbedarf haben.

[0010] Zur Lösung der Aufgabenstellung sind die Merkmale des Anspruchs 1 vorgesehen.

[0011] Demnach wird jeder Datenblock sukzessive entschachtelt, indem Entschachtelungs-Zieladressen zunächst nur für einen vorbestimmten Abschnitt des Datenblocks berechnet werden, nachfolgend eine Entschachtelung des entsprechenden Abschnitts des erhaltenen (verschachtelten) Datenblocks durchgeführt wird, und dieser Vorgang so oft wiederholt wird, bis der gesamte Datenblock Abschnitt für Abschnitt entschachtelt ist. Auf diese Weise wird der Speicherplatzbedarf bei der Entschachtelung wesentlich reduziert, weil lediglich die Entschachtelungs-Zieladressen von Datenblockabschnitten und nicht des gesamten Datenblocks gespeichert werden müssen.

[0012] Demzufolge kennzeichnet sich eine vorteilhafte Ausgestaltung der Erfindung dadurch, daß die in dem nächsten Schritt berechneten Entschachtelungs-Zieladressen unter Überschreiben der zuvor in dem Zieladressenspeicher abgelegten Zieladressen abgespeichert werden.

[0013] Eine weitere vorteilhafte Ausgestaltung des erfindungsgemäßen Verfahrens kennzeichnet sich dadurch, daß die Verschachtelung nach mehreren unterschiedlichen Verschachtelungsvorschriften durchführbar ist, daß eine Erzeugungsregel zur Berechnung der unterschiedlichen Verschachtelungsvorschriften vorgegeben ist, und daß die Vorausberechnung der Entschachtelungs-Zieladressen bezüglich der Abschnitte von Datensymbolen unmittelbar aus der Erzeugungsregel ohne vorhergehende Berechnung von Zieladressen für die Verschachtelung vorgenommen wird. Aufgrund des Wegfalls der Berechnung der Verschachtelungs-Zieladressen wird eine weitere Verminderung des Speicherplatzbedarfs für die Entschachtelung erreicht.

[0014] Bei der Erzeugungsregel kann es sich um den UMTS-Standard TS 25.212 handeln, welcher für jede Datenblocklänge K eine Turbo-Verschachtelungsvorschrift in Form einer Koordinaten-Transformationsmatrix bestehend aus R Zeilen und C Spalten definiert. In diesem Fall kann jeder der vorgegebenen Abschnitte eines Datenblocks eine Anzahl von nz · C aufeinanderfolgenden Datensymbolen des verschachtelten Datensignals aufweisen, wobei nz eine ganze Zahl gleich oder größer 1 ist.

[0015] Vorzugsweise ist nz = 1, d. h. es wird eine zeilenweise Turbo-Entschachtelung des Datenblocks vorgenommen.

[0016] Nachfolgend wird die Erfindung anhand eines die Turbo-Entschachtelung (gemäß UMTS-Standard TS 25.212) betreffenden Ausführungsbeispiels unter Bezugnahme auf die Zeichnung näher erläutert; in dieser zeigt:

[0017] Fig. 1 ein Blockschaltbild eines bekannten Turbo-Codierers zur Erzeugung eines Turbo-Codes;

[0018] Fig. 2 ein Blockschaltbild eines bekannten Turbo-Decodierers zur Decodierung eines Turbo-codierten Empfangssignals;

[0019] Fig. 3 eine Darstellung zur Erläuterung einer Verschachtelungs-Permutationsmatrix und der inversen Permutationsmatrix sowie des erfindungsgemäßen Prinzips der abschnittsweisen Turbo-Entschachtelung;

[0020] Fig. 4 eine Darstellung zur Erläuterung der Intra-Zeilen Permutation bei der Erzeugung einer Verschachtelungs-Transformationsmatrix für K = 3840 im UMTS-Standard; und

[0021] Fig. 5 eine Darstellung entsprechend Fig. 4, in welcher die Hintereinanderausführung von zwei Koordinatentransformationen zur Realisierung der Intra-Zeilen Permutation und einer Koordinatentransformation zur Realisierung der Inter-Zeilen Permutation für die Erzeugung der UMTS-Verschachtelungs-Transformationsmatrix dargestellt sind.

[0022] Fig. 1 zeigt beispielhaft das Blockschaltbild eines Turbo-Codierers TCOD, wie er in einem UMTS-Sender zur Erzeugung eines Turbo-codierten Datensignals D eingesetzt werden kann. Im Rahmen der Erfindung können auch andere Turbo-Codierer verwendet werden.

[0023] Der Turbo-Codierer TCOD weist einen Turbo-Verschachteler IL, zwei identische, rekursive, systematische Faltungscodierer RSC1 und RSC2 (z. B. 8-Zustands-Faltungscodierer), zwei optionale Punktierer PKT1 und PKT2 und einen Multiplexer MUX auf.

[0024] Die Aufgabe des Turbo-Codierers TCOD besteht darin, einem digitalen Eingabesignal X zur Fehlerschutzcodierung Redundanz hinzuzufügen. Das Eingabesignal besteht aus einer Folge von Datensymbolen, z. B. Bits. Bei dem digitalen Eingabesignal X kann es sich beispielsweise um ein quellencodiertes Sprach- oder Videosignal handeln.

[0025] Der Turbo-Codierer TCOD erzeugt ein digitales Ausgabesignal D, das durch Multiplexen des Eingabesignals X (sogenanntes systematisches Signal), eines von RSC1 codierten und ggf. von PKT1 punktierten Signals Y1 und eines von IL verschachtelten, von RSC2 codierten und ggf. von PKT2 punktierten Signals Y2 erzeugt wird.

[0026] Der Turbo-Verschachteler IL führt eine blockweise Verschachtelung des Eingabesignals X durch. Das heißt, daß der Turbo-Verschachteler IL in ständiger Wiederholung jeweils K Datensymbole (K ist eine ganze, positive Zahl und bezeichnet die Datenblocklänge) entgegennimmt, umsortiert und in geänderter Reihenfolge wieder ausgibt. Das Umsortieren (Permutieren) der Datensymbole erfolgt nach einer für eine konstante Datenblocklänge K immer gleichen Vorschrift.

[0027] Im UMTS-Standard ist die Blocklänge K variabel und liegt zwischen 40 und 5114 Bits. Wie später noch näher erläutert wird, ist für jede Datenblocklänge im Standard eine spezielle Verschachtelungsvorschrift vorgeschrieben.

[0028] Das fehlerschutzcodierte Datensignals D wird dann in geeigneter Weise auf einen Träger moduliert und über einen Übertragungskanal (z. B. Mobilfunkkanal) übertragen.

[0029] Die Decodierung eines Turbo-codierten Empfangssignals in einem Empfänger wird nachfolgend unter Bezugnahme auf den in Fig. 2 gezeigten, bekannten Turbo-Decodierer TDEC erläutert. Auch andere Bauweisen von Turbo-Decodierern sind möglich und können zur Durchführung des erfindungsgemäßen Verfahrens eingesetzt werden.

[0030] Der Turbo-Decodierer TDEC umfaßt einen ersten und einen zweiten Demultiplexer DMUX1 und DMUX2, einen Speicher MEM, einen ersten und zweiten Faltungsdecodierer DEC1 und DEC2, einen Turbo-Verschachteler IL', einen ersten und einen zweiten Turbo-Entschachteler DIL1 und DIL2 sowie eine Entscheidungslogik (Schwellenwertentscheider) TL.

[0031] Von einem Demodulator (nicht dargestellt) des Empfängers wird eine entzerrte Datenfolge $\hat{D}$ bereitgestellt, die die im Empfänger rekonstruierte codierte Datenfolge D ist.

[0032] Die Funktionsweise des in Fig. 2 gezeigten Turbo-Decodierers TDEC wird im folgenden kurz erläutert.

[0033] Der erste Demultiplexer DMUX1 spaltet das entzerrte Datensignal $\hat{D}$ in das entzerrte systematische Datensignal $\hat{X}$ (rekonstruierte Version des Eingabesignals X) und ein entzerrtes Redundanzsignal $\hat{Y}$ auf. Letzteres wird von dem zweiten Demultiplexer DMUX2 in die beiden entzerrten Redundanz-Teilsignale $\hat{Y}1$ und $\hat{Y}2$ (rekonstruierte Versionen der Redundanz-Teilsignale Y1 und Y2) aufgespalten.

[0034] Die beiden Faltungsdecodierer DEC1 und DEC2 können z. B. MAP-Symbolschätzer sein. Der erste Faltungsdecodierer DEC1 berechnet ausgehend von den Datensignalen $\hat{X}$ und $\hat{Y}1$ und einem Rückkoppelsignal Z logarithmische Zuverlässigkeitsdaten $\Lambda 1$ in Form von LLRs (Log-Likelihood Ratios).

[0035] Die Zuverlässigkeitsdaten $\Lambda 1$ werden von dem Turbo-Verschachteler IL' verschachtelt und die verschachtelten Zuverlässigkeitsdaten $\Lambda 1_I$ werden dem zweiten Faltungsdecodierer DEC2 zugeführt. Die Arbeitsweisen der Turbo-Verschachteler IL und IL' sind identisch. Der zweite Faltungsdecodierer DEC2 berechnet aus den verschachtelten Zuverlässigkeitsdaten $\Lambda 1_I$ und aus den rekonstruierten Redundanz-Teilsignaldaten $\hat{Y}2$, die in dem Speicher MEM bereitgehalten werden, ein verschachteltes Rückkoppelsignal Z_I und verschachtelte zweite logarithmische Zuverlässigkeitsdaten $\Lambda 2_I$, ebenfalls in Form von LLR's.

[0036] Das verschachtelte Rückkoppelsignal Z_I wird von dem ersten Turbo-Entschachteler DIL1 entschachtelt und ergibt das Rückkoppelsignal Z.

[0037] Die dargestellte Rekursionsschleife wird mehrmals (z. B. 5 Mal) durchlaufen. Jedem Durchlauf liegen die Daten desselben Datenblocks zugrunde. Die beim letzten Durchlauf erhaltenen verschachtelten zweiten Zuverlässigkeitsdaten $\Lambda 2_I$ werden von dem zweiten Entschachteler DIL2 entschachtelt und als entschachtelte Zuverlässigkeitsdaten $\Lambda 2$ der Entscheidungslogik TL zugeführt. Diese bestimmt daraufhin ein Datensignal E(X), welches eine Folge von Schätzwerten für die Datensymbole des Eingabesignals X ist.

[0038] Nach der Turbo-Decodierung eines Datenblocks und Ausgabe der entsprechenden Folge von Schätzwerten E(X) wird der nächste Datenblock Turbo-decodiert.

[0039] Eine detaillierte Beschreibung der Arbeitsweise eines Turbo-Decodierers ist in dem Kapitel E.3.3 "Rekursive MAP-Symbolschätzung" des genannten Buchs von P. Jung auf den Seiten 353 bis 361 angegeben, die hiermit zum Inhalt dieser Schrift wird.

[0040] Wie beispielhaft an dem in Fig. 2 dargestellten Turbo-Decodierer TDEC ersichtlich, umfaßt eine Turbo-Decodierung bei jedem Schleifendurchlauf eine Turbo-Verschachtelungsprozedur (IL') und eine Turbo-Entschachtelungsprozedur (DIL1) sowie eine abschließende Turbo-Entschachtelungsprozedur (DIL2). Die beiden Turbo-Entschachtelungsprozeduren sind identisch.

[0041] Die Verschachtelungsvorschrift kann mathematisch durch eine Permutation beschrieben werden. Die Permuta-

tion ordnet jeder Ausgangs- oder Quellenadresse eindeutig eine Zieladresse für die Umsortierung der Datensymbole eines Datenblocks zu. Quellenadresse ist die ursprüngliche Stelle des Datensymbols im Datenblock und die Zieladressen ist die Stelle des umsortierten Datensymbols im verschachtelten Datenblock.

[0042] In **Fig.** 3 wird das der Erfindung zugrundeliegende Prinzip anhand eines einfachen Beispiels erläutert.

[0043] Zunächst wird der Verschachtelungsvorgang betrachtet. Eine einen Datenblock bildende Datenfolge bestehend aus K = 9 Datensymbolen {a, b, c, d, e, f, g, h, i} soll verschachtelt werden. **Fig.** 3, oberer Teil, zeigt einen als 3 × 3-Speicherzellen-Matrix dargestellten Verschachtelungs-Eingabedatenspeicher V_iDS, einen als 3 × 3-Speicherzellen-Matrix dargestellten Verschachtelungs-Ausgabedatenspeicher V_fDS und eine 3 × 3-Permutationsmatrix P, deren Elemente ebenfalls in einem Speicher (Zieladressenspeicher) abgelegt sind.

[0044] Die Datenfolge wird in den Verschachtelungs-Eingabedatenspeicher V_iDS eingelesen und dort, wie in **Fig.** 3 dargestellt, in Zeilenrichtung abgespeichert.

[0045] Die Speicherplätze der Datenspeicher V_iDS und V_fDS sind in Zeilenrichtung mit Adressen n = 1 bis 9 durchnumeriert. Die Adressen n sind im rechten oberen Eck der jeweiligen Speicherzellen eingetragen.

[0046] Die Permutationsmatrix P gibt für das im Verschachtelungs-Eingabedatenspeicher V_iDS in der Speicherzelle mit Adresse n abgespeicherte Datensymbol die Verschachtelungs-Zieladresse V-Adr(n) im Verschachtelungs-Ausgabedatenspeicher V_fDS an. Beim Verschachteln wird demnach das in V_iDS auf Speicherplatz **1** abgespeicherte Datensymbol, nämlich a, in V_fDS auf dem Speicherplatz **3** abgespeichert, das in V_iDS auf Speicherplatz **2** abgespeicherte Datensymbol, nämlich b, wird in V_fDS auf dem Speicherplatz **7** abgespeichert, . . usw. Das Auslesen des Verschachtelungs-Ausgabedatenspeichers V_fDS erfolgt ebenfalls in Zeilenrichtung, d. h. die verschachtelte Datenfolge lautet {g, e, a, c, h, f, b, i, d}.

[0047] Das Entschachteln wird gemäß **Fig.** 3, unterer Teil, analog dem Verschachteln, jedoch unter Verwendung der inversen Permutationsmatrix (der Begriff invers bezieht sich auf die Operation der Nacheinanderausführung von Permutationen), bezeichnet als P^{-1}, durchgeführt. Die inverse Permutationsmatrix P^{-1} ist in **Fig.** 3 angegeben. Die Elemente der inversen Permutationsmatrix sind in einem Entschachtelungs-Zieladressenspeicher gespeichert.

[0048] Es wird nun angenommen, daß der Verschachteler in der Lage sein soll, eine Vielzahl von unterschiedlichen Verschachtelungsvorschriften auszuführen. Dabei sollen die diversen Verschachtelungsvorschriften nicht in Form einer Vielzahl von im Verschachteler abgespeicherten Permutationsmatrizen bereitgehalten werden, sondern es wird im folgenden vorausgesetzt, daß eine spezielle Erzeugungsregel existiert, mit der die verschiedenen Permutationsmatrizen in Abhängigkeit von einem oder mehreren Erzeugungsparametern (z. B. der Datenblocklänge K) aufgebaut werden können. Wie im folgenden noch dargelegt, sind diese Voraussetzungen bei der Turbo-Verschachtelung gemäß dem UMTS-Standard erfüllt.

[0049] Die herkömmliche Vorgehensweise zur Durchführung der Entschachtelung ist die folgende: Zunächst wird gemäß der Erzeugungsregel die gewünschte Permutationsmatrix P vollständig (d. h. sämtliche Verschachtelungs-Zieladressen) berechnet. Dann wird die vollständig berechnete Permutationsmatrix P invertiert. Mittels der invertierten Permutationsmatrix P^{-1} wird dann die Entschachtelung vorgenommen.

[0050] Das erfindungsgemäße Vorgehen bei der Turbo-Entschachtelung unterscheidet sich von der herkömmlichen Vorgehensweise dadurch, daß zunächst nur ein bestimmter, vorgegebener Abschnitt der inversen Permutationsmatrix P^{-1}, z. B. die in der ersten Zeile angegebenen Entschachtelungs-Zieladressen E-Adr(n) = 7, 5, 1, (schraffiert dargestellt) für n = 1, 2, 3 bestimmt werden. Anschließend, d. h. vor der Bestimmung weiterer Entschachtelungs-Zieladressen, wird eine erste teilweise Entschachtelung des verschachtelten Datensignals vorgenommen. Dabei werden die ersten drei Datensymbole g, e, a der verschachtelten Datenfolge, die in dem Entschachtelungs-Eingabedatenspeicher E_iDS (entspricht V_fDS) auf den ersten drei Speicherplätzen abgespeichert sind, in die Speicherplätze **7**, **5**, **1** des Entschachtelungs-Ausgabedatenspeichers E_fDS geschrieben. Anschließend wird ein weiterer vorgegebener Abschnitt der inversen Permutationsmatrix P^{-1}, z. B. die in der zweiten Zeile angegebenen Entschachtelungs-Zieladressen **3**, **8**, **6**, berechnet. Dann wird wiederum der diesbezügliche Schreibvorgang durchgeführt. Diese Vorgehensweise wird solange fortgesetzt, bis die verschachtelte Datenfolge vollständig entschachtelt ist.

[0051] Mit anderen Worten werden weder die Permutationsmatrix P noch aus dieser die inverse Permutationsmatrix P^{-1} vollständig berechnet, sondern es werden immer nur jeweils die Matrixelemente (Entschachtelungs-Zieladressen) der inversen Permutationsmatrix P^{-1} berechnet, die für die Entschachtelung des vorgegebenen Abschnitts des verschachtelten Datenblocks gerade benötigt werden. Vorteilhaft ist dabei der geringe Speicherplatzbedarf zum Abspeichern der Entschachtelungs-Zieladressen im Entschachtelungs-Zieladressenspeicher, da in jedem Entschachtelungsschritt die im vorhergehenden Entschachtelungsschritt verwendeten Zieladressen überschrieben werden können. Bei dem erläuterten Beispiel (d. h. bei einer zeilenweisen Entschachtelung) muß der Entschachtelungs-Zieladressenspeicher nicht neun sondern nur drei Speicherzellen umfassen.

[0052] Es wird bemerkt, daß die Möglichkeit, die Verschachtelungs-Permutationsmatrix P mittels der Erzeugungsregel gezielt für bestimmte, vorgegebene Abschnitte berechnen zu können, nicht impliziert, daß auch eine abschnittsweise Berechnung der inversen Permutationsmatrix P^{-1} bezüglich vorgebbarer Abschnitte möglich ist. Berechnet man z. B. die Verschachtelungs-Zieladressen der ersten Zeile der Permutationsmatrix P, erhält man die Werte 3, 7, 4. Mit diesen Werten lassen sich die Adressen **1**, **2**, **3** der inversen Permutationsmatrix P^{-1} (punktiert dargestellt) berechnen, die jedoch nicht zur Entschachtelung eines vorgegebenen Abschnitts von Datensymbolen, z. B. der in der ersten Zeile des Speichers E_iDS abgespeicherten Datensymbole, ausreichen. Dieses Beispiel macht deutlich, daß selbst dann, wenn eine abschnittsweise Berechnung der Permutationsmatrix P möglich sein sollte, für die Berechnung eines vorgegebenen Abschnitts der inversen Permutationsmatrix P^{-1} zunächst im allgemeinen die vollständige Permutationsmatrix P zu berechnen ist.

[0053] Im folgenden wird für den Fall des UMTS-Standards eine Möglichkeit der partiellen Berechnung der inversen Permutationsmatrix P^{-1} (Entschachtelungs-Zieladressenmatrix) angegeben. Die Erkenntnis, daß eine solche abschnittsweise Berechnung der inversen Permutationsmatrix P^{-1} im UMTS-Standard möglich ist, ist Teil der Erfindung.

[0054] Wie bereits erwähnt, ist in dem UMTS-Standard eine Erzeugungsregel angegeben, mittels der für jede mögliche

Blocklänge K eine spezielle [redacted]achtelungsvorschrift generiert werden kann. Jede V[redacted]chtelungsvorschrift ist in Form einer Koordinatentrans[redacted]ation zwischen dem Entschachtelungs-Eingabedatensp[redacted]her E_iDS und dem Entschachtelungs-Ausgabedatenspeicher E_fDS angegeben.

[0055] Zum besseren Verständnis der Erfindung wird nachfolgend zunächst die im UMTS-Standard TS 25.212 V3.3.0 vereinbarte Erzeugungsregel zur Bestimmung der zugehörigen Koordinaten-Transformationsmatrix wiedergegeben. Die Koordinaten-Transformationsmatrix enthält die gleiche Information wie die anhand **Fig. 3** erläuterte Permutationsmatrix, unterscheidet sich von dieser jedoch dadurch, daß die Permutationsvorschrift in Form einer zweidimensionalen Koordinatentransformation (und nicht einer eindimensionalen Zieladressen-Zuweisungsvorschrift) dargestellt ist.

1. Schritt (Definition der Transformationsmatrix)

1.1 Definition der Anzahl R der Zeilen:
R = 5, falls K = 40 bis 159 Bits (Fall 1)
R = 10, falls K = 160 bis 200 Bits oder K = 481 bis 530 Bits (Fall 2)
R = 20, andernfalls (Fall 3)
1.2 Definition der Anzahl C der Spalten:
Fall 2, für K = 481 bis 530 Bits: C = p = 53 sonst:

(i) Suche der minimalen Primzahl p, so daß $0 \leq (p + 1) - K/R$
(ii) falls $0 \leq p - K/R$, dann gehe zu (iii) sonst: $C = p + 1$
(iii) falls $0 \leq p - 1 - K/R$, dann: $C = p - 1$ sonst: $C = p$

1.3 Die Eingabedatensequenz wird dann Zeile für Zeile in eine RxC-Eingabedaten-Speichermatrix (entspricht V_iDS) geschrieben.

2. Schritt (Intra-Zeilen Permutation)

Fall A: C = p

(A1) Auswahl einer primitiven Wurzel g aus der folgenden Tabelle:

p	g	p	g	p	g	p	g	p	g
7	3	47	5	101	2	157	5	223	3
11	2	53	2	103	5	163	2	227	2
13	2	59	2	107	2	167	5	229	6
17	3	61	2	109	6	173	2	233	3
19	2	67	2	113	3	179	2	239	7
23	5	71	7	127	3	181	2	241	7
29	2	73	5	131	2	191	19	251	6
31	3	79	3	137	3	193	5	257	3
37	2	83	2	139	2	197	2		
41	6	89	3	149	2	199	3		
43	3	97	5	151	6	211	2		

(A2) Konstruktion einer Basissequenz c(i) für die Intra-Zeilen Permutation nach:

$c(i) = [g \cdot c(i - 1)] \text{mod} p$, $i = 1, 2, \ldots, (p - 2)$ $c(0) = 1$

wobei mod die Modulo-Operation bezeichnet.
(A3) Suche nach dem Satz $\{q_j\}$ der minimalen Primzahlen, $j = 1, 2, \ldots, R - 1$, mit:
- $ggT\{q_j, p - 1\} = 1$ (ggT = größter gemeinsamer Teiler)
- $q_j > 6$
- $q_j > q_{j-1}$
- $q_0 = 1$

(A4) Der Satz $\{g_j\}$ wird permutiert, der durch die Permutation erhaltene Satz mit $\{p_j\}$ bezeichnet, die Permutationsvorschrift lautet:

$p_{PX(j)} = q_j$, $j = 0, 1 \ldots, R - 1$,

wobei $P_X(j)$ die Inter-Zeilen Permutation ist, die im dritten Schritt definiert wird.

(A5) Durchführen der j-ten Intra-Zeilen Permutation, j = 0, 1, . . ., R – 1, nach:

$c_j(i) = c([i \cdot p_j] \bmod (p-1))$, i = 0, 1, 2, . ., (p – 2) und $c_j(p-1) = 0$,

wobei $c_j(i)$ die Position des Eingabebits des i-ten Ausgangs nach der Permutation der j-ten Zeile ist.

Fall B: C = p + 1

(B1) Wie Fall A1
(B2) Wie Fall A2
(B3) Wie Fall A3
(B4) Wie Fall A4
(B5) Durchführen der j-ten Intra-Zeilen Permutation, j = 0, 1, . . ., R – 1, nach:

$c_j(i) = c([i \cdot p_j] \bmod (p-1))$, i = 0, 1, 2, . ., (p – 2) und $c_j(p-1) = 0$, und $c_j(p) = p$,

(B6) Falls K = C · R, dann tausche $c_{R-1}(p)$ gegen $c_{R-1}(0)$ aus, wobei $c_j(i)$ die Position des Eingabebits des i-ten Ausgangs nach der Permutation der j-ten Zeile ist.

Fall C: C = p – 1

(C1) Wie Fall A1
(C2) Wie Fall A2
(C3) Wie Fall A3
(C4) Wie Fall A4
(C5) Durchführen der j-ten Intra-Zeilen Permutation, j = 0, 1, . . ., R – 1, nach:
$c_j(i) = c([i \cdot p_j] \bmod (p-1)) - 1$, i = 0, 1, 2, . ., (p – 2), wobei $c_j(i)$ die Position des Eingabebits des i-ten Ausgangs nach der Permutation der j-ten Zeile ist.

3. Schritt (Inter-Zeilen Permutation)

[0056] Durchführen der Inter-Zeilen Permutation $P_X(j)$, j = 0, 1, . ., R – 1, X = A, B, C oder D, nach den folgenden Schemata, wobei $P_X(j)$ die ursprüngliche Position der j-ten permutierten Zeile ist:

P_A: {19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 10, 8, 13, 17, 3, 1, 16, 6, 15, 11} für R = 20

P_B: {19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 16, 13, 17, 15, 3, 1, 6, 11, 8, 10} für R = 20

P_C: {9, 8, 7, 6, 5, 4, 3, 2, 1, 0} für R = 10

P_D: {4, 3, 2, 1, 0} für R = 5

[0057] Die verschiedenen Schemata werden folgendermaßen eingesetzt:

Blocklänge K	$P_X(j)$
40-159 Bits	P_D
160-200 Bits	P_C
201-480 Bits	P_A
481-530 Bits	P_C
531-2280 Bits	P_A
2281-2480 Bits	P_B
2481-3160 Bits	P_A
3161-3210 Bits	P_B
3211-5114 Bits	P_A

mit X = A oder B oder C oder D

[0058] Fig. 4 erläutert anhand eines Beispiels für K = 3840 den Aufbau der Transformationsmatrix. Dabei werden die einzelnen Elemente der Matrix anhand ihrer Zeilen-Spalten Koordinaten (j, i) identifiziert und die durch den obigen Standard vorgeschriebene Koordinaten-Transformationen betrachtet.

[0059] Gemäß den Definitionen unter den Punkten 1.1 und 1.2 ergibt sich C = 192 (Anzahl der Spalten) und R = 20 (Anzahl der Zeilen). Die minimale Primzahl lautet p = 191.

[0060] Für den 2. Schritt gi[redacted]B. Gemäß dem Schritt B1 wird die primitive Wurzel g[redacted] = 191 bestimmt. Es ergibt sich g = 19.

[0061] In Schritt B2 wird die Basissequenz c(i) berechnet. Die berechneten Werte c(i) sind in **Fig. 4** in dem fett umrandeten horizontalen Bereich angegeben.

[0062]. In dem Schritt B3 wird der Satz $\{q_j\}$ der minimalen Primzahlen, j = 0 bis R – 1, berechnet. Der Satz der minimalen Primzahlen lautet:

{1, 7, 11, 13, 17, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83}

[0063] Im folgenden Beispiel wird der Schritt B5 mit dem Satz $\{g_j\}$ der minimalen Primzahlen durchgeführt; der Übergang auf den Satz der permutierten minimalen Primzahlen erfolgt erst nachfolgend bei der Inter-Zeilen Permutation. Somit wird für die j-te Zeile die zugehörige Intra-Zeilen Permutation nach der Gleichung $c_j(i) = c([i \cdot q_j] \bmod(p-1))$ berechnet. Die Intra-Zeilen Permutationsvorschrift $c_j(i)$ würde, sofern sie gesondert an den Datensymbolen im Verschachtelungs-Eingabedatenspeicher V_iDS ausgeführt würde (was nicht der Fall ist, da sie lediglich zum Aufbau der Transformationsmatrix dient), bewirken, daß ein in dem Verschachtelungs-Eingabedatenspeicher V_iDS auf die Zeilen-Spalten Koordinate (j, i) eingelesenes Datensymbol auf eine Speicherzelle eines (fiktiven) Zwischenspeichers mit der Zeilen-Spalten Koordinate $(j, c_j(i))$ abgespeichert würde.

[0064] Die Intra-Zeilen Permutation $c_j(i)$ hängt von dem Zeilenindex j ab, d. h. ist für jede Zeile unterschiedlich.

[0065] Gemäß der im Schritt B5 angegebenen Gleichung kann die Intra-Zeilen Permutation als Nacheinanderausführung einer "inneren" Intra-Zeilen Permutation nach der Vorschrift

$$c\_in_j(i) = [i \cdot q_j] \bmod(p-1)$$

und einer "äußeren" Intra-Zeilen Permutation nach der Vorschrift

$$c\_out(i) = c(i)$$

durchgeführt werden.

[0066] Die innere Intra-Zeilen Permutation $c\_in_j(i)$ ist für jede Zeile unterschiedlich, während die äußere Intra-Zeilen Permutation c_out(i) für sämtliche Zeilen identisch ist.

[0067] Einige der bei der inneren Intra-Zeilen Permutation erhaltenen Spalten-Zielkoordinatenwerte $c\_in_j(i)$ sind in der in **Fig. 4** dargestellten R×C-Transformationsmatrix eingetragen. Für i = 1 (1-te Spalte) ergibt sich die Sequenz $\{q_j\}$ der minimalen Primzahlen.

[0068] Die Werte in der Spalte i = 4 sind in der **Fig. 4** fett umrandet. Sie berechnen sich gemäß der Gleichung $c\_in_j(4) = [4 \cdot q_j] \bmod 190$.

[0069] Für die Speicherzelle mit der Koordinate i = 4, j = 3 ergibt sich der Wert $c\_in_3(4) = [4 \cdot 13] \bmod 190 = 52$.

[0070] Die innere Intra-Zeilen Permutation, die also die Koordinate (3, 4) auf die Koordinate (3, 52) abbildet, wird in **Fig. 5** durch den Pfeil A veranschaulicht.

[0071] Die äußere Intra-Zeilen Permutation c_out(i) wird durch den Pfeil B dargestellt. Die Zielkoordinate (3, 52) der inneren Intra-Zeilen Permutation, die die Ausgangskoordinate für die äußere Intra-Zeilen Permutation ist, wird auf die Zielkoordinate (3, 86) der äußeren Intra-Zeilen Permutation abgebildet (die damit auch die Zielkoordinate der gesamten Intra-Zeilen Permutation ist).

[0072] Im Ergebnis folgt für dieses Beispiel:

$$c_3(4) = c\_out(c\_in_3(4)) = 86$$

[0073] In dem 3. Schritt wird die Inter-Zeilen Permutation gemäß dem Schema P_A ausgeführt. Da $P_A(j = 3) = 4$, wird die Zielkoordinate (3, 86) der Intra-Zeilen Permutation auf die Zielkoordinate (4, 86) der Inter-Zeilen Permutation abgebildet. Die Inter-Zeilen Permutation entspricht einem (tatsächlich nicht stattfindenden) Transfer des in der Speicherzelle des fiktiven Zwischenspeichers mit der Koordinate (3, 86) abgespeicherten Datensymbols in die Speicherzelle des Verschachtelungs-Ausgabedatenspeichers V_fDS mit der Koordinate (4, 86). Die Inter-Zeilen Permutation wird in **Fig. 5** durch den Pfeil C veranschaulicht.

[0074] Allgemein ergibt sich für die Verschachtelung somit die UMTT-Koordinaten-Abbildungsvorschrift:

$$(j, i) \rightarrow (P_X(j), c_j(i))$$

[0075] Aus der Koordinaten-Abbildungsvorschrift der Transformationsmatrix lassen sich die eindimensionalen Verschachtelungs-Zieladressen der Permutationsmatrix P gemäß der folgenden Beziehung berechnen:

Quellenadresse: $n = j \cdot C + i$

Verschachtelungs-Zieladresse: $V\text{-}Adr(n) = P_X(j) \cdot C + c_j(i)$

[0076] Damit ist die Permutationsmatrix gemäß **Fig. 3**, oberer Teil, berechenbar.

[0077] Für das Beispiel ergibt sich:

Quellenadresse: $n = 3 \cdot 192 + 4 = 580$

Verschachtelungs-Zieladresse: $V\text{-}Adr(579) = 4 \cdot 192 + 86 = 854$

[0078] Das heißt, in der Permutationsmatrix P steht in dem Feld mit der Adresse n = 580 (entspricht der Koordinate (3, 4)) der Verschachtelungs-Zieladressenwert 854.

[0079] Im folgenden wird erläutert, wie gemäß der Erfindung die Entschachtelungs-Zieladressen der ersten Zeile der inversen Permutationsmatrix P^{-1} berechnet werden können, ohne zuvor eine Berechnung der Permutationsmatrix P

durchführen zu müssen.

[0080] Zunächst wird die Zeilen- und Spaltenzahl der inversen Permutationsmatrix P^{-1} bestimmt. Die Bestimmung erfolgt gemäß dem Schritt 1, das heißt ist identisch mit der Bestimmung der Zeilen- und Spaltenzahl der Permutationsmatrix P.

[0081] Die Koordinaten der inversen Permutationsmatrix P^{-1} werden in der Form (j, i), das heißt ebenfalls als Zeilen-Spalten-Koordinaten, angegeben.

[0082] Zunächst wird die inverse Abbildung der im UMTS-Standard unter Schritt 3 definierten Inter-Zeilen Permutation ausgeführt. Die inversen Inter-Zeilen Permutationen $P_X^{-1}(j)$, $j = 0, 1, .., R - 1$, lauten für die Fälle X = A, B, C oder D:

P^{-1}_A: {4, 15, 5, 14, 3, 6, 17, 7, 11, 1, 10, 19, 8, 12, 2, 18, 16, 13, 9, 0} für R = 20
P^{-1}_B: {4, 15, 5, 14, 3, 6, 16, 7, 18, 1, 19, 17, 8, 11, 2, 13, 10, 12, 9, 0} für R = 20
P^{-1}_C: {9, 8, 7, 6, 5, 4, 3, 2, 1, 0} für R = 10
P^{-1}_D: {4, 3, 2, 1, 0} für R = 5

[0083] Die Auswahl X = A, B, C oder D der inversen Inter-Zeilen Permutation folgt aus dem unter Schritt 3 angegebenen Schema.

[0084] Die Koordinaten-Abbildungsvorschrift der inversen Inter-Zeilen Permutation ist:

$$(j, i) \rightarrow (P_X^{-1}(j), i)$$

[0085] Dabei ist (j, i) die Ausgangskoordinate einer Speicherzelle des Entschachtelungs-Eingabedatenspeichers E_iDS.

[0086] In einem nächsten Schritt wird die Zeilen-Koordinate durch aufeinanderfolgendes Ausführen der Umkehrabbildungen der äußeren Intra-Zeilen Permutation und der inneren Intra-Zeilen Permutation berechnet.

[0087] Die Koordinatentransformation bezüglich der inversen äußeren Intra-Zeilen Permutation lautet:

$$(P_X^{-1}(j), i) \rightarrow (P_X^{-1}(j), c\_out^{-1}(i))$$

[0088] Dabei bezeichnet $c\_out^{-1}(i) = c^{-1}(i)$ die inverse äußere Intra-Zeilen Permutation.

[0089] In einem letzten Koordinaten-Transformationsschritt wird die inverse innere Intra-Zeilen Permutation ausgeführt. Die entsprechende Abbildungsvorschrift lautet:

$$(P_X^{-1}(j), c\_out^{-1}(i)) \rightarrow (P_X^{-1}(j), c\_in_{P_X^{-1}(j)}^{-1}(c\_out^{-1}(i)))$$

[0090] Die Berechnung der Entschachtelungs-Zieladressen E-Adr(n) erfolgt dann unter Verwendung der nachstehenden Abkürzungen

$$d_i = c\_in_{P_X^{-1}(j)}^{-1}(c\_out^{-1}(i))$$

$$d_j = P_X^{-1}(j)$$

gemäß der Gleichung

$$\text{E-Adr}(n) = d_j \cdot C + d_i$$

wobei $n = j \cdot C + i$ die Quellen-Adresse des in dem Entschachtelungs-Eingabedatenspeicher E_iDS abgespeicherten verschachtelten Datensignals ist.

[0091] Ausnahmen dieses Entschachtelungsschemas treten bei den Spalten p – 1 und p für den Fall C = p + 1 und für die Spalte p – 1 für den Fall C = p auf.

[0092] Diese Spalten werden nicht der Intra-Zeilen Permutation unterworfen, das heißt bei ihnen wird weder die äußere Intra-Zeilen Permutation noch die innere Intra-Zeilen Permutation vorgenommen. Deshalb beschränkt sich der Entschachtelungsablauf auf die Umkehrung der Inter-Zeilen Permutation.

[0093] Bei der Entschachtelung wird bezüglich der Spalte p lediglich die Inter-Zeilen Permutation durchgeführt. Das Ergebnis der Entschachtelung lautet daher:

$d_i = i = p$

$d_j = P_X^{-1}(j)$, X = A, B, C oder D

[0094] Die Spalte p – 1 wird bei der Verschachtelung der Inter-Zeilen Permutation unterzogen und dann auf die Spalte 0 abgebildet. Das Ergebnis des Entschachtelungsablaufs ist daher:

$d_i = p - 1$ für $i = 0$

$d_j = P_X^{-1}(j)$ X = A oder B oder C oder D

[0095] Die Berechnung der Entschachtelungs-Zieladressen erfolgt auch hier gemäß der bereits angegebenen Formel $\text{E-Adr}(n) = d_j \cdot C + d_i$

[0096] Bei der Durchführung des Entschachtelungsschrittes werden nun zunächst die Ziel-Entschachtelungsadressen

E-Adr(n) für einen bestimmten vorgegebenen Abschnitt des Entschachtelungs-Eingabespeichers E_iDS, z. B. in dem beschriebenen Beispiel die Entschachtelungsadressen n = 0, 1, . ., 191 für eine bestimmte Zeile j, berechnet. Hierzu müssen zu dem Zeilenindex j der Zeilenkoordinatenwert d_j und sämtliche Spaltenkoordinatenwerte d_i berechnet werden. Die berechneten Entschachtelungs-Zieladressen E-Adr(n) für die Zeile j werden im Zieladressenspeicher abgespeichert. Dieser muß zu diesem Zweck beim betrachteten Beispiel lediglich 192 Speicherzellen, allgemein maximal 256 Speicherzellen, einer Wortbreite von z. B. 13 Bits aufweisen. Bei einem aus mehreren Zeilen bestehenden Datenblockabschnitt ist der Zieladressenspeicher entsprechend größer auszulegen.

[0097] Mittels dieser 192 Zieladressen wird dann eine Entschachtelung der ersten 192 Datensymbole des verschachtelten Datensignals durchgeführt. Der Ablauf entspricht der anhand **Fig.** 3 erläuterten Vorgehensweise.

[0098] Nachdem die ersten 192 Datensymbole (bzw. ein anderer frei wählbarer Abschnitt des verschachtelten Datenblocks) entschachtelt ist, wird der nächste Satz von Entschachtelungs-Zieladressenwerten E-Adr(n) berechnet, und es wird gemäß der berechneten Entschachtelungs-Zieladressenwerte die Umspeicherung der Datensymbole des zweiten betrachteten Abschnitts des Datenblocks durchgeführt. Sofern eine zeilenweise Entschachtelung ausgeführt wird, ist der Datenblock nach R = 5 oder R = 10 oder R = 20 alternierenden Entschachtelungs-Zieladressen-Berechnungsschritten, Entschachtelungs-Zieladressen-Abspeicherungsschritten, wobei die bei der vorhergehenden Prozedur verwendeten Zieladressen überschrieben werden, und Datensymbol-Umspeicherschritten vollständig entschachtelt. Das beschriebene Verfahren der abschnittsweisen oder sequentiellen Entschachtelung eines blockweise verschachtelten Datensignals wurde anhand der Turbo-Entschachtelung gemäß dem UMTS Standard erläutert, ist jedoch nicht auf diese Bedingungen beschränkt, sondern kann allgemein als Entschachtelungsprozedur für blockweise verschachtelte Datensignale eingesetzt werden.

Patentansprüche

1. Verfahren zur Entschachtelung eines gemäß einer vorgegebenen Verschachtelungsvorschrift blockweise verschachtelten Datensignals, wobei ein Datenblock K Datensymbole umfaßt, **dadurch gekennzeichnet,**
daß Entschachtelungs-Zieladressen (E-Adr(n)) bezüglich eines ersten vorgegebenen Abschnitts der K zu entschachtelnden Datensymbole eines Datenblocks berechnet und in einem Zieladressenspeicher abgelegt werden;
daß dieser Abschnitt der Datensymbole gemäß den im Zieladressenspeicher abgelegten Entschachtelungs-Zieladressen entschachtelt wird;
daß in einem nächsten Schritt neue Entschachtelungs-Zieladressen für einen nächsten vorgegebenen Abschnitt der K zu entschachtelnden Datensymbole dieses Datenblocks berechnet und in dem Zieladressenspeicher abgelegt werden;
daß der nächste Abschnitt der Datensymbole gemäß den im Zieladressenspeicher abgelegten neuen Entschachtelungs-Zieladressen entschachtelt wird; und
daß auf diese Weise der gesamte Datenblock abschnittsweise entschachtelt wird.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß in dem nächsten Schritt berechnete Entschachtelungs-Zieladressen unter Überschreiben der zuvor in dem Zieladressenspeicher abgelegten Entschachtelungs-Zieladressen abgespeichert werden.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet,
daß die Verschachtelung nach mehreren unterschiedlichen Verschachtelungsvorschriften durchführbar ist,
daß eine Erzeugungsregel zur Berechnung der unterschiedlichen Verschachtelungsvorschriften vorgegeben ist, und
daß die Vorausberechnung der Entschachtelungs-Zieladressen bezüglich der Abschnitte von Datensymbolen unmittelbar aus der Erzeugungsregel ohne eine vorherige Berechnung der Zieladressen für die Verschachtelung vorgenommen wird.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, daß es sich bei der Entschachtelung um eine Turbo-Entschachtelung, d. h. eine Entschachtelung eines mit einem Turbo-Verschachteler verschachtelten Datensignals, handelt.

5. Verfahren nach Anspruch 3, dadurch gekennzeichnet,
daß es sich bei der vorgegebenen Erzeugungsregel um den UMTS-Standard TS 25.212 handelt, welcher für jede Datenblocklänge K eine Verschachtelungsvorschrift in Form einer Koordinaten-Transformationsmatrix bestehend aus R Zeilen und C Spalten definiert, und
daß jeder der vorgegebenen Abschnitte eine Anzahl von nz · C aufeinanderfolgende Datensymbole des verschachtelten Datensignals aufweist, wobei nz eine ganze Zahl gleich oder größer 1 ist.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, daß nz = 1.

Hierzu 4 Seite(n) Zeichnungen

- Leerseite -

Nummer: DE 100 48 872 A1
Int. Cl.[7]: H 03 M 13/29
Offenlegungs 25. April 2002

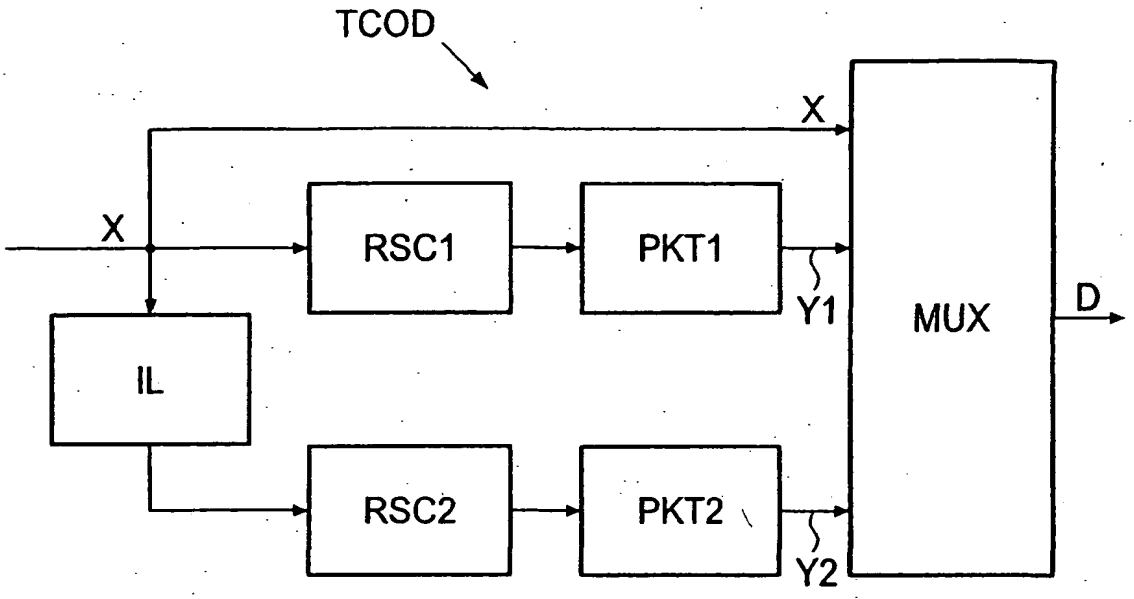


Fig.1
Stand der Technik

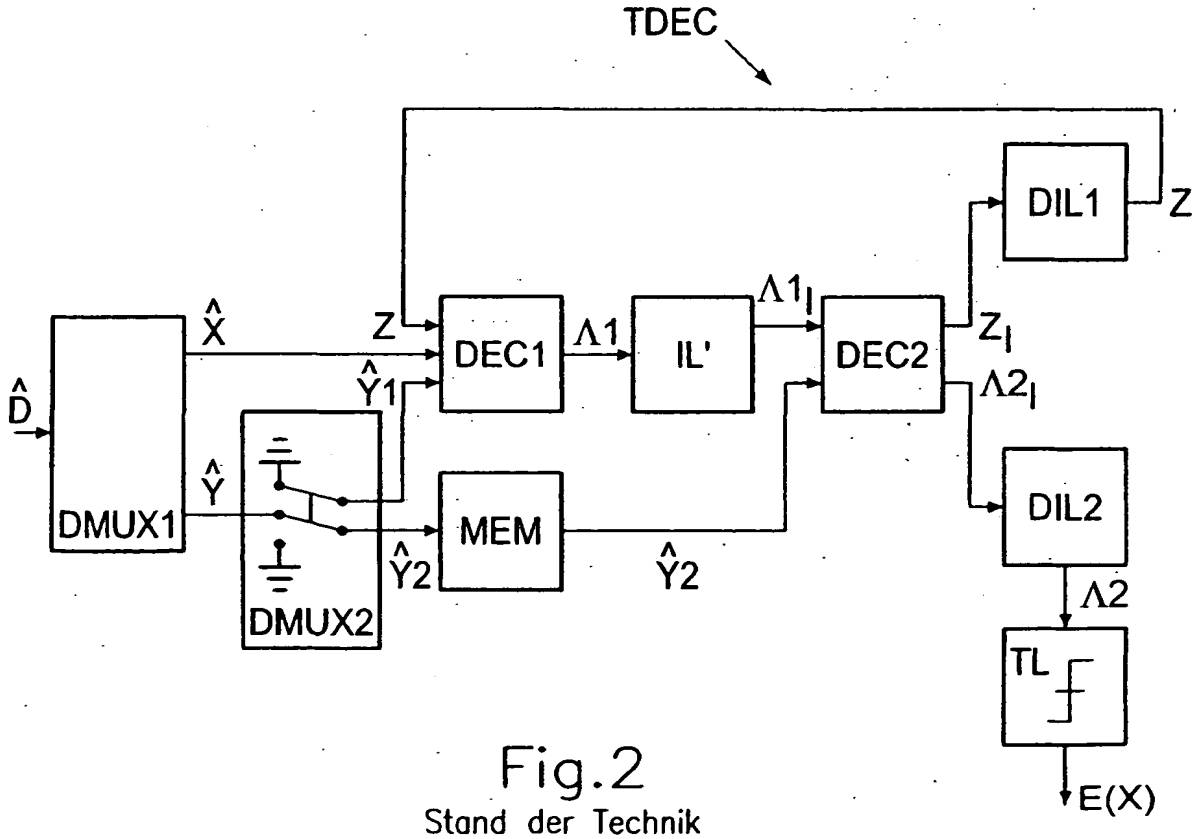


Fig.2
Stand der Technik

Nummer: DE 100 48 872 A1
Int. Cl.[7]: H03M 13/29
Offenlegungstag: 25. April 2002

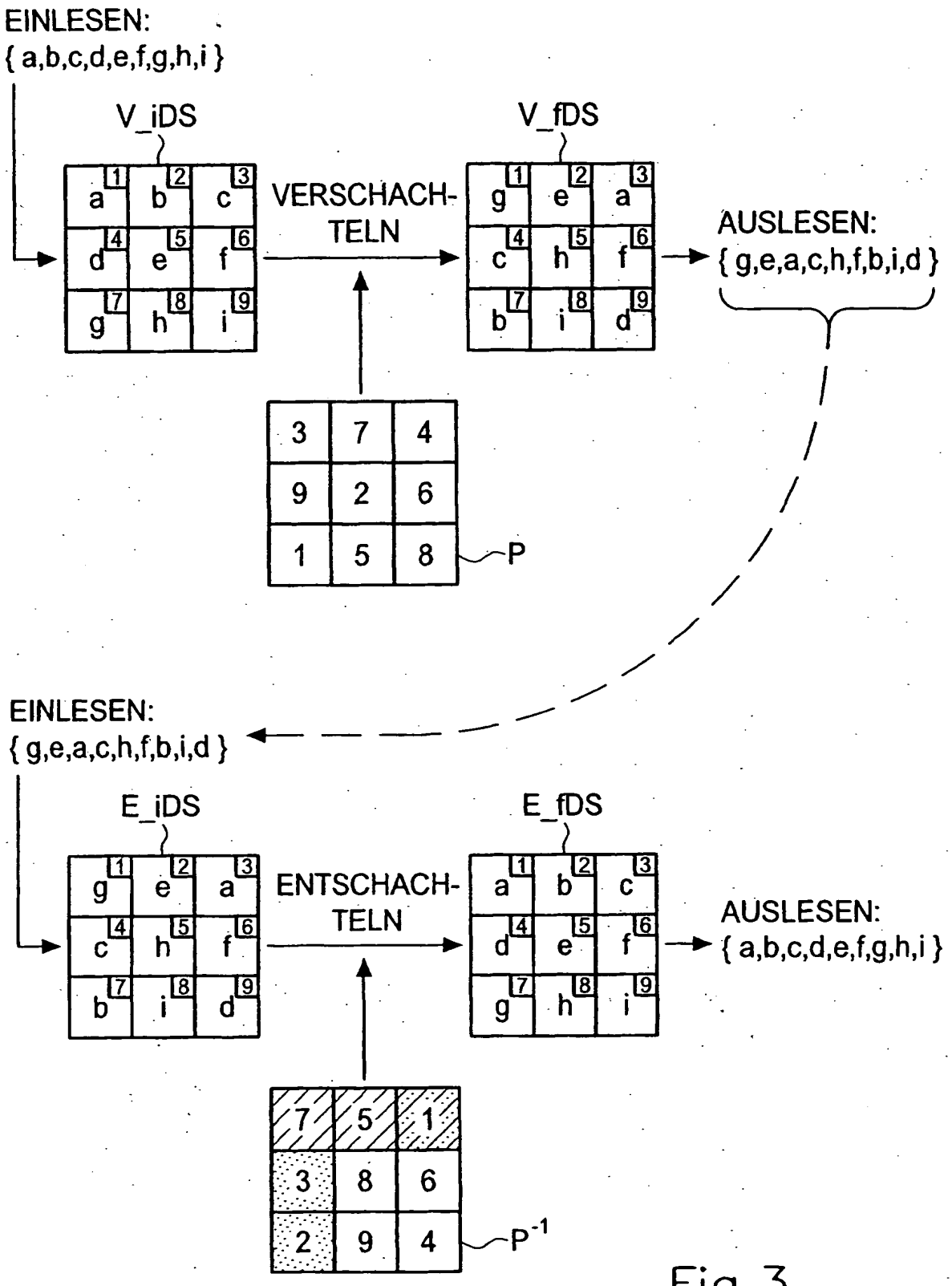


Fig.3

Nummer: DE 100 48 872 A1
Int. Cl.[7]: H 03 M 13/29
Offenlegung: 25. April 2002

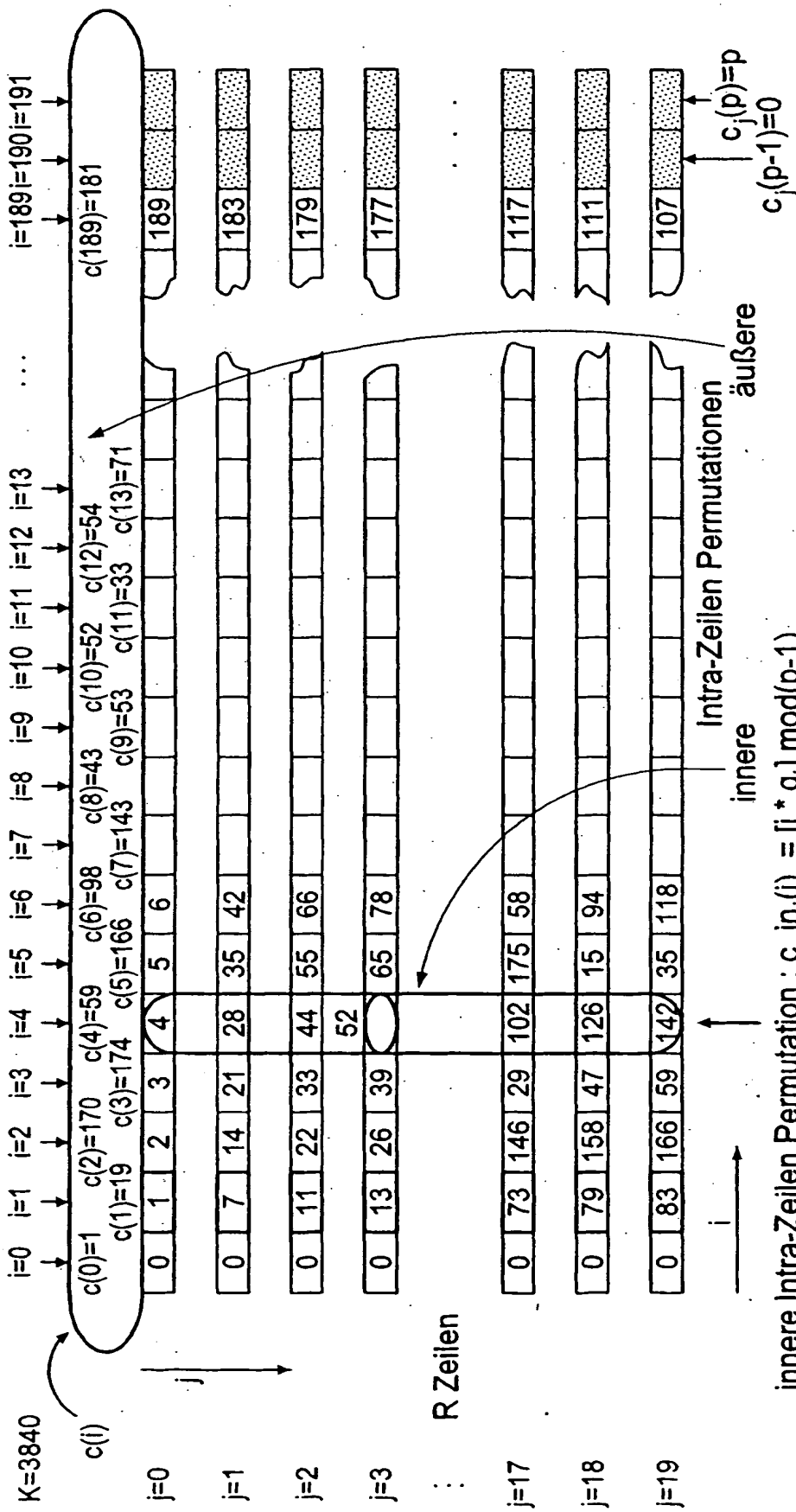


Fig. 4

Nummer: DE 100 48 872 A1
Int. Cl.[7]: H03M 13/29
Offenlegungstag: 25. April 2002

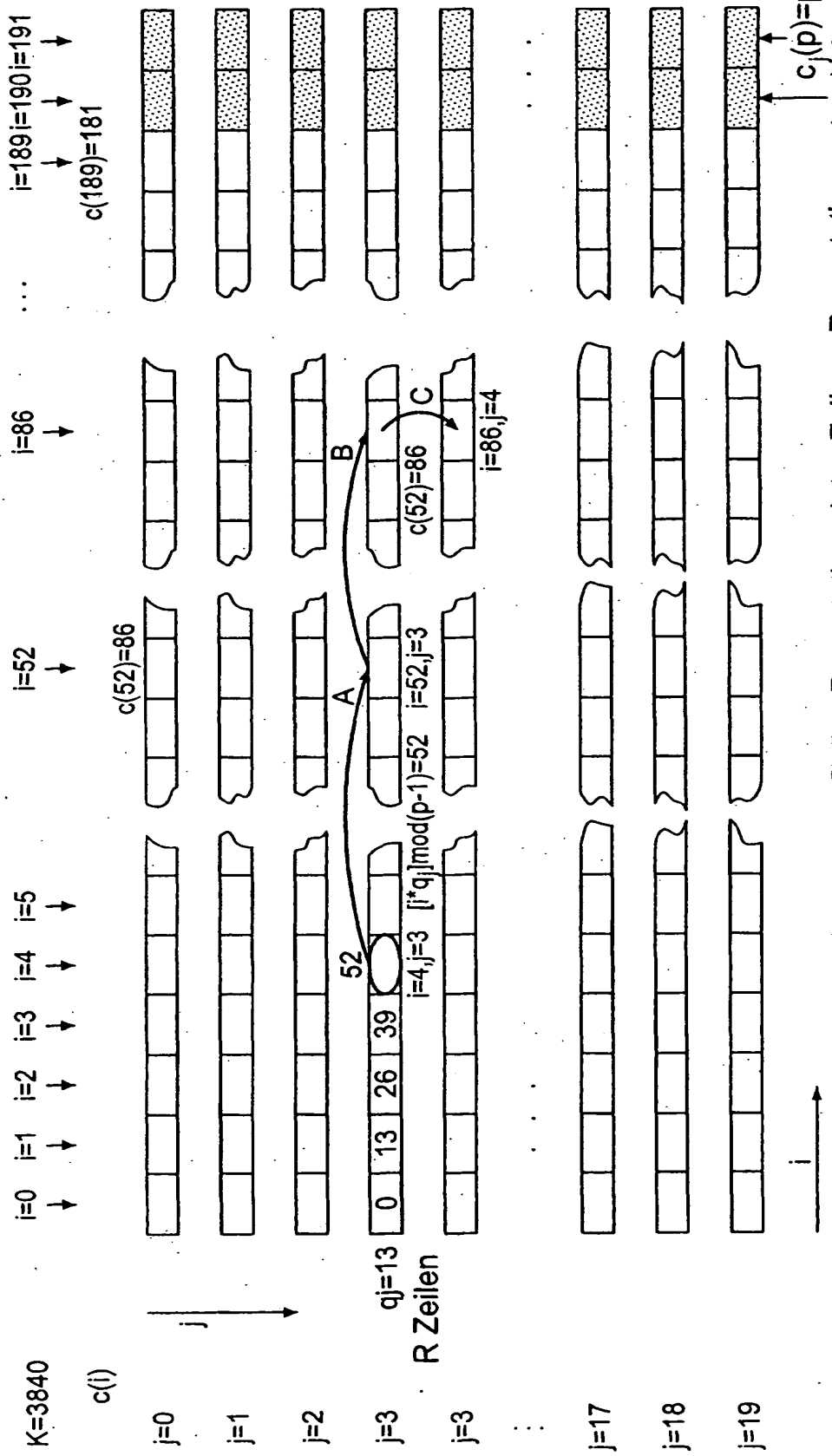


Fig.5